FPGA Co-Processors for Next-Generation Performance Monitoring Hardware

Jeffrey Kulick*
Brett Boren
Department of Electrical and Computer Engineering University of Alabama in Huntsville
Huntsville, Al. 35899

Today, performance-monitoring hardware consists of counters tied to discrete processor events such as instructions retired or branch mis-predicts. Future performance analysis efforts will require performance monitoring hardware to be much more cognizant of the structure of a program's execution and be able to correlate multiple program events. For example, at UAH we are studying the use of co-processors for real-time analysis of instruction traces for software intrusion detection. Future performance monitoring hardware must also be able to encode and compress multiple event transactions to reduce the workload that traditional performance hardware introduces. Current performance hardware must be sampled at every event of interest, which creates significant added workload for fine grain observations such as basic block traversals. Lastly, performance monitoring hardware currently has no ability to trace control-flow within an executing process. The enhancements described below allow for a small coprocessor to act as an independent tracing unit that can gather data with low execution overhead on a fine-grained basis and offload that data independently of the main processor.

The co-processor is geared towards tracing control-flow of a process. A GNU C compiler has been modified to emit tags at user selected events of interest such as loop entry, exit, continuation, and function entry/exit. The co-processor detects these tags and forms compressed records of multiple traversals of basic blocks and larger program paths. Because these tags are emitted by compiler generated code, no control-flow recognition logic is needed in hardware calling for a simpler co-processor. The goal is to have the co-processor complete its work within one basic block execution (on average) thus not delaying the main processor Additionally, since the tracing is based on the compiler's control flow graph, compiler optimizations have no effect on the data accuracy.

This coprocessor is being implemented on a Xilinx Virtex-II FPGA within the Microblaze soft processor system as a peripheral. The main processor uses memory writes to communicate events to the coprocessor. The coprocessor can then independently offload the recorded data into main memory. The uClinux variant of Linux has been used to perform operating system support functionality such as uploading completed traces to an external ftp server.

Unless the actual issued instructions have augmented tag bits, an instrumented process will never run as fast as its un-instrumented precursor. However by moving more of trace processing logic into hardware a much less intrusive and higher performance solution can be achieved. Such gains may allow for a future where all programs are traced allowing for several applications including software fault detection or dynamic hardware/software reconfiguration.

*kulick@ece.uah.edu